УДК 534.843:004.9 DOI: https://doi.org/10.15407/jai2020.03.014

***О.І. Белей, К.К. Колесник***
Національний університет «Львівська політехніка», Україна
вул. Митрополита Андрея, 5, м. Львів, 79015

# МОДЕЛЮВАННЯ СИСТЕМИ ВИЯВЛЕННЯ АТАК НА ОСНОВІ ГІБРИДИЗАЦІЇ БІНАРНИХ КЛАСИФІКАТОРІВ

***O.I. Belej, K.K. Kolesnyk***
Lviv Polytechnic National University, Ukraine
5, Mytropolyt Andrei St., Lviv, 79015

# MODELING OF ATTACK DETECTION SYSTEM BASED ON HYBRIDIZATION OF BINARY CLASSIFIERS

**Анотація.** В дослідженні розглядаються питання розробки методики виявлення аномальних мережевих з’єднань на основі гібридизації методів обчислювального інтелекту. Виконано аналіз підходів до виявлення аномалій і зловживань в комп’ютерних мережах. У рамках цього аналізу запропоновано класифікацію методів виявлення мережевих атак. Основні результати зводяться до побудови багатокласових моделей, які дозволяють підвищити ефективність функціонування системи виявлення атак, і можуть використовуватися для побудови систем класифікації мережевих параметрів під час атак. Розроблено модель штучної імунної системи на базі еволюційного підходу, алгоритм генетико-конкурентного навчання мережі Кохонена та методику ієрархічної гібридизації бінарних класифікаторів з додатком до виявлення аномальних мережевих з'єднань. Розроблено архітектуру мережевої розподіленої системи виявлення атак. Архітектура системи виявлення атак є дворівневою: перший рівень забезпечує первинний аналіз окремих пакетів і мережевих з'єднань за допомогою сигнатурного аналізу, на другому рівні здійснюється обробка агрегованих мережевих потоків даних за допомогою адаптивних класифікаторів. Проведено сигнатурний аналіз для дослідження швидкодії мережі на основі алгоритмів Ахо-Корасік і Бойера-Мура й реалізовані їх поліпшені аналоги за допомогою технологій OpenMP і CUDA. Наведена архітектура та показано основні моменти функціонування генератора мережевих атак. Розроблено систему генерації мережевих атак. Дана система складається з двох компонентів: асинхронного прозорого проксі-сервера TCP-сесій і frontend-інтерфейсу генератора мережевих атак. Результати експериментів підтвердили те, що функціональні й нефункціональні вимоги, а також вимоги, що пред’являються до обчислювальних інтелектуальних систем, виконуються для розробленої системи виявлення атак.

**Ключові слова:** гібридизація; аномалії; мережеві з’єднання; бінарні класифікатори; евристичні підходи; штучний інтелект; виявлення атак

**Abstract.** The study considers the development of methods for detecting anomalous network connections based on hybridization of computational intelligence methods. An analysis of approaches to detecting anomalies and abuses in computer networks. In the framework of this analysis, a classification of methods for detecting network attacks is proposed. The main results are reduced to the construction of multi-class models that increase the efficiency of the attack detection system, and can be used to build systems for classifying network parameters during the attack. A model of an artificial immune system based on an evolutionary approach, an algorithm for genetic-competitive learning of the Kohonen network and a method of hierarchical hybridization of binary classifiers with the addition to the detection of anomalous network connections have been developed. The architecture of the network distributed attack detection system has been developed. The architecture of the attack detection system is two-tier: the first level provides the primary analysis of individual packets and network connections using signature analysis, the second level processes the processing of aggregate network data streams using adaptive classifiers. A signature analysis was performed to study network performance based on the Aho-Korasik and Boyer-Moore algorithms and their improved analogues were implemented using OpenMP and CUDA technologies. The architecture is presented and the main points of operation of the network attack generator are shown. A system for generating network attacks has been developed. This system consists of two components: an asynchronous transparent proxy server for TCP sessions and a frontend interface for a network attack generator. The results of the experiments confirmed that the functional and non-functional requirements, as well as the requirements for computing intelligent systems, are met for the developed attack detection system.

**Keywords:** hybridization; anomalies; network connections; binary classifiers; heuristic approaches; artificial intelligence; attack detection

## Вступ

Розробка системи виявлення атак є одним із пріоритетних напрямків у галузі інформаційної безпеки. Важливість вирішення цього завдання обумовлюється постійним збільшенням і різноманітністю комп'ютерних мережевих загроз, реалізація яких може призводити до серйозних фінансових втрат у різних організаціях. Подібне зростання атакуючих дій з кожним роком потребує затрат значно більших сил і тимчасових витрат з боку адміністраторів і аналітиків безпеки. У компаніях, залучених до виробництва критично важливої продукції, для підтримки безпеки корпоративних мережевих ресурсів витрачаються великі фінансові та матеріальні кошти, спрямовані на утримання спеціального обладнання у вигляді компонентів системи виявлення атак і обслуговуючого його персоналу. Для забезпечення коректної інтерпретації переданих у пакетах даних необхідно виконувати їх складання в мінімальний логічний потік – мережеве з'єднання, що дозволить оперувати більш високорівневими характеристиками мережевого трафіку для виявлення аномалій, властивих мережевому й транспортному рівням моделі OSI.

Для виявлення мережевих атак можуть застосовуватися як сигнатурні механізми пошуку шаблонних аномальних дій, так і евристичні підходи. У разі сигнатур, вирішення задачі зводиться до реалізації процедури, яка виконує перевірку входження заданої байтової послідовності всередині вмісту мережевих пакетів. Недоліками такого рішення є складність створення репрезентативного набору з подібними записами й обмеження у виявленні модифікованих варіантів відомої атаки. Навпаки, евристичні підходи дозволяють виявляти приховані закономірності в аналізованих мережевих потоках. Саме ця особливість пояснює їх широку популярність у науково-дослідному співтоваристві й відіграє ключову роль при виборі й проєктуванні ядра системи виявлення атак. З іншого боку, в основі функціонування більшості комерційних відкритих програмних рішень переважає підхід, який базується на сигнатурному зіставленні зі зразком і характеризується мінімальним числом помилкових спрацьовувань. Для збереження переваг обох підходів використовується прийом їх комбінування, що, як і раніше, залишаються неповною мірою дослідженими. Тому завдання виявлення аномальних мережевих з'єднань є актуальним, а пропонований у цьому дослідженні модельно-методичний апарат, який використовує гібридизацію різнорідних методів обчислювального інтелекту й сигнатурного аналізу, спрямований на його вирішення. Сфера обчислювального інтелекту охоплює дослідження біологічно інспірованих моделей, спрямованих на обробку низькорівневих даних про об'єкт без використання експертних знань. Під терміном «гібридизація» розуміється комбінування різнорідних вирішень в єдину систему класифікації об'єктів.

## Постановка проблеми

Аналіз робіт у цій галузі показав, що для виявлення мережевих атак відсутня гнучка методика навчання колективу адаптивних бінарних класифікаторів, і більшість досліджень обмежується розглядом лише однієї схеми комбінування вирішень. Тому дослідження спрямоване на розробку узагальненого підходу до побудови універсальної структури для зберігання та подання класифікаторів – дерева класифікаторів та алгоритму каскадного навчання його вузлів, що дозволить, у контексті виявлення аномалій мережевих з'єднань, поєднувати різнорідні модулі без суворої прив'язки, агрегувати їх виходи й підвищити ефективність системи виявлення атак за рахунок можливості вибору найкращої схеми комбінування вирішень.

Дослідження полягає в розробленні компонентів, призначених для підвищення коректності детектування мережевих атак, що дозволить забезпечити необхідний рівень захищеності інформаційних ресурсів. Використання запропонованої методики гібридизації бінарних класифікаторів надасть можливість об'єднати різнорідні засоби виявлення мережевих атак для створення гібридної системи виявлення атак. За рахунок використання детекто-

рів як мінімальної одиниці класифікації мережевих атак, проєктування системи виявлення атак здійснюється знизу-вгору: спочатку її ядро пристосовується до виявлення індивідуальних типів атак, а потім – формує правила для порівняння підозрілого з'єднання з відповідним класом. Розроблений модельно-методичний апарат може бути використаний як для захисту комп'ютерних мереж, так і для вирішення інших, більш загальних задач, пов'язаних з класифікацією об'єктів.

**Аналіз останніх досліджень і публікацій**

Серед перших робіт, що містять передумови до постановки та вирішення задачі виявлення аномалій, можна вважати [1]. У [2] автор передбачає, що зловмисник може бути виявлений за допомогою аналізу вмісту журналу аудиту контрольованої системи й наявністю відхилень, витягнутих з нього записів, від встановлених адміністратором. У [3] пропонує розглянути статистичну модель, що складається з шести компонентів: (1) користувач, процес, система; (2) файли, програми, команди, пристрої; (3) записи журналу аудиту, що представляють собою результат дії суб'єктів над об'єктами; (4) шаблони поведінки суб'єктів; (5) записи, що генеруються при виявленні аномальної поведінки; і (6) правила, які визначають умови їх спрацьовування й наступні дії.

Варто зазначити, що побудова шаблону нормальної поведінки є трудомістким завданням і часто не завжди здійсненним. Так, на практиці виявляється, що не кожна аномальна поведінка є атакою [4]. Зокрема, адміністратор мережі може застосовувати налагоджувальні утиліти, такі як ping, traceroute, mtr для діагностики мережного оточення. Дії такого роду не мають будь-яких нелегальних намірів, проте системи виявлення аномалій розпізнають цю діяльність як властиву нелегітимній мережевій активності.

Однією з класичних і фундаментальних робіт, присвячених виявленню зловживань, є [5]. Для вирішення цього завдання автори даної роботи пропонують використовувати розфарбовані мережі Петрі.

У обчислювальному інтелекті такий інтелектуальний агент являє собою систему, яка здатна функціонувати розумно [6]. На відміну від інших галузей ШІ, напрямок ОІ значною мірою акцентується на побудові й дослідженні обчислювальних моделей. ОІ був уведений з метою відокремити загін теоретичних дисциплін, які на той момент вже стали цілком самостійними й розвиненими. Серед них були виділені нейронні мережі, генетичні алгоритми, нечіткі системи, еволюційне програмування й активне життя. У [7] визначено наступні характеристики, властиві для обчислювально-інтелектуальної системи: (1) здатність обробляти числові дані низького рівня; (2) наявність компонентів розпізнавання образів; (3) відсутність необхідності базуватися на експертних знаннях. Крім того, виділяється ряд додаткових вимог до систем подібного класу: (1) обчислювальна адаптивність; (2) стійкість у разі наявності шумів; (3) висока швидкість функціонування; (4) рівень помилок, наближений до людської діяльності.

У дослідженні [8] представлена система динамічного забезпечення та моніторингу ресурсів, мультиагентна система для управління ресурсами хмарного провайдера з урахуванням вимог до якості обслуговування клієнтів, визначених угодою про рівень обслуговування. А дослідження [9] розширює пропозицію стандарту хмарних обчислювальних інтерфейсів для відповідного багатокласового обчислювального агента.

У попередніх дослідженнях [10] авторами розглядається недосконалість існуючих методів виявлення вторгнень, а також мінливий характер зловмисних дій на комп'ютерні системи. На основі проведеного аналізу була запропонована гібридна схема виявлення та класифікації мережевих атак на основі комбінації адаптивних класифікаторів.

**Мета дослідження**

Полягає в підвищенні ефективності функціонування системи виявлення атак за допомогою модельно-методичного апарату, який базується на підході гібридизації обчислювального інтелекту.

**Виклад основного матеріалу**

Загальне уявлення пропонованої методики для виявлення аномальних мережевих з'єднань показано на рис. 1 і складається з наступних п'яти етапів:

– побудова дерева класифікаторів;
– формування параметрів мережевих з'єднань;
– попередня обробка параметрів мережевих з'єднань;
– ієрархічний обхід дерева класифікаторів у ширину;
– виявлення аномальних мережевих з'єднань.

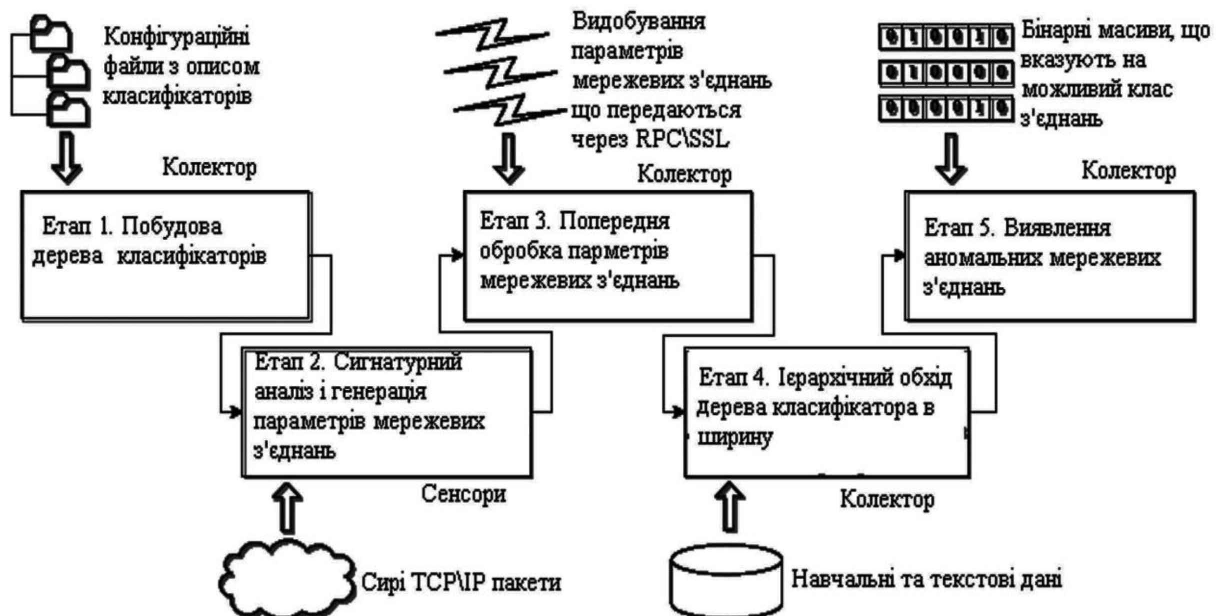



Рис. 1. Основні етапи методики ієрархічної гібридизації бінарних класифікаторів для виявлення аномальних мережевих з'єднань

Перший етап методики може бути охарактеризований як підготовчий, він включає в себе вибір структури окремих бінарних класифікаторів: розмірності й числа шарів, параметрів і алгоритмів навчання, типів функцій активації, функцій приналежності й ядерних функцій. Для кожного детектора може бути складений набір навчальних правил. Ставлячи різну сукупність таких наборів правил, можна сформувати групу детекторів.

Детектори всередині кожної такої групи об'єднуються в класифікатор на основі підходів один-до-всіх (one-vs-all), один-до-одного (one-vs-one) або їх різних похідних варіацій. У першому підході кожен детектор $F_{jk}^{(i)}: R^n \to \{0,1\}(k = 1,..., m)$ навчається на даних $\{(x_i, [\bar{c}_l = k])\}_{l=1}^{M}$ і функціонування групи детекторів описується наступним чином:

$$F_j^{(i)}(z) = \begin{cases} \{0\}, \\ \{k \mid F_{jk}^{(i)}(z) = 1\}_{k=1}^{m} \end{cases} \qquad (1)$$

У другому підході кожний з +1 = детекторів 0 1 навчається на безлічі об'єктів, що належать тільки двом класам з мітками 0 і $k_1$ $-\{(x_l, 0 \mid \bar{c}_l = k_0)\}_{l=1}^{M} \bigcup \{(x_l, 1 \mid \bar{c}_l = k_1)\}_{l=1}^{M}$, де $0 \le k_0 < k_1 \le m$. Однією з похідних варіацій попередніх підходів для комбінування детекторів може бути згадане класифікаційне бінарне дерево [2]. Формально така структура задається рекурсивно наступним чином:

$$F_j^{(i)} = \left\{ \arg \max_{\bar{c} \in \{0,...,m\}} \sum_{k=\bar{c}+1}^{m} [F_{j\bar{c}k}^{(i)}(z) = 0] + \sum_{k=0}^{c-1} [F_{jk\bar{c}}^{(i)}(z) = 1] \right\} \qquad (2)$$
.

Тому функціонування групи детекторів, представлених у вигляді вузлів такого дерева, описується за допомогою ре-

курсивної функції, яка задає послідовну дихотомію множини:

$$F_j^{(i)} = \phi_j^{(i)}(\mu, z) \quad (3)$$

$$F_j^{(i)}(z) = \begin{cases} \mu, \\ \phi_j^{(i)}(L_\mu, z), \\ \phi_j^{(i)}(R_\mu, z) \end{cases} \quad (4)$$

Застосування функції до вихідного набору міток класів і об'єктів, що класифікуються, дозволяє здійснювати однозначний пошук мітки класу цього об'єкта. Це пояснюється тим, що, оскільки по мірі спуску вниз класифікаційним деревом відбувається диз'юнктивне розбиття множини міток класів, то після досягнення й спрацьовування термінального детектора залишається тільки одна можлива мітка для класифікації вхідного об'єкта як вихідного результату. Тому для класифікаційного дерева неможливі конфліктні випадки при класифікації об'єктів, які можуть мати місце для двох інших підходів комбінування.

Іншим підходом є спрямований ациклічний граф, який організовує детектори у зв'язну динамічну структуру, яка може бути задана наступною формулою:

$$DAG_\mu = \begin{cases} \left\langle F_{j\mu k_0 k_1}^{(1)}, DAG_{\mu \setminus \{k_0\}}, DAG_{\mu \setminus \{k_1\}} \right\rangle, \\ \mu \end{cases} \quad (5)$$

У іншому випадку виключається мітка 0. Процес повторюється до тих пір, поки множина не вироджується в одноелементну. Нами наведено характеристики розглянутих схем об'єднання детекторів у багатокласову модель, призначену для співвіднесення вхідного об'єкта однієї або з деякими (+1) мітками класів. На рис. 2 зображено класифікаційне бінарне дерево й спрямований ациклічний граф (рис. 3).

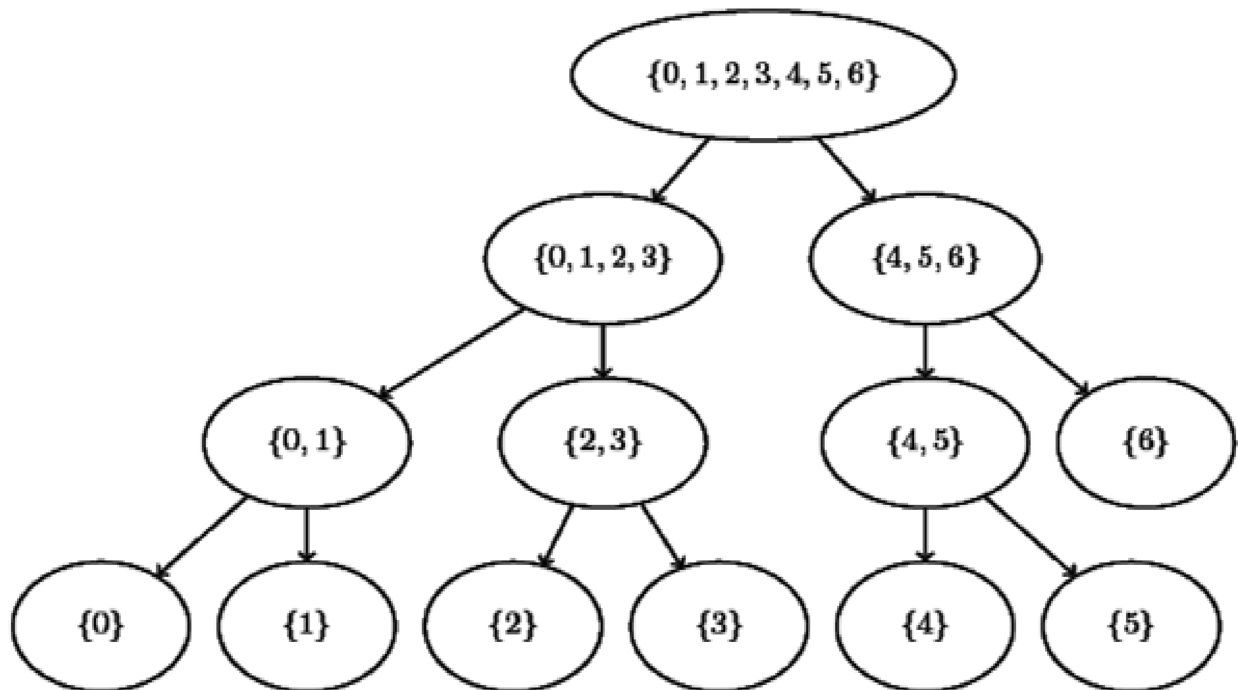



Рис. 2. Класифікаційне бінарне дерево

З розглянутих чотирьох схем тільки одна, а саме класифікаційне бінарне дерево, має змінне число детекторів, які можуть використовуватися в процесі класифікації об'єктів. Мінімальне значення досягається, коли активується детектор, розташований в корені дерева й навчений для розпізнавання тільки одного класу об'єктів серед усіх інших.

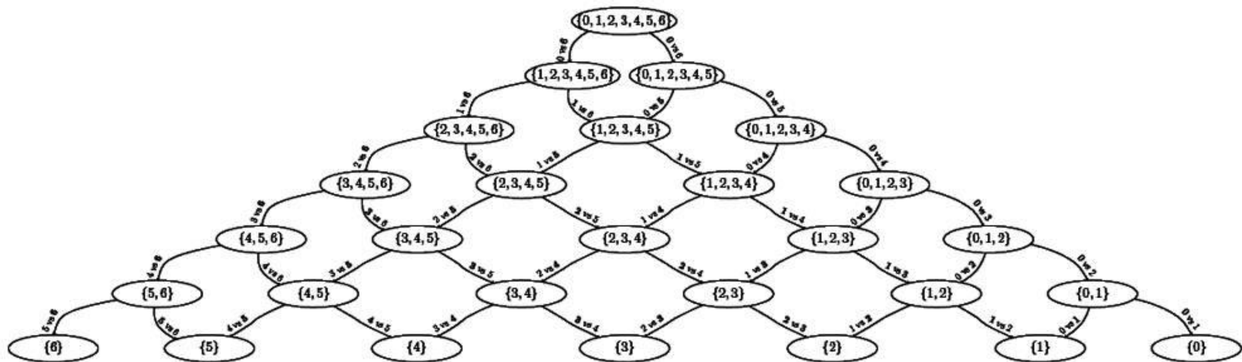



Рис. 3. Спрямування ациклічного графа

Максимальне значення досягається, коли дерево представляється послідовним списком і активується найбільш віддалений у ньому детектор. На рис. 4 представлений класифікатор, у якому кожна з груп детекторів навчається на різних випадкових бутстрап-підвибірках, які можуть включати повторювані впорядковані елементи з вихідного навчального набору.

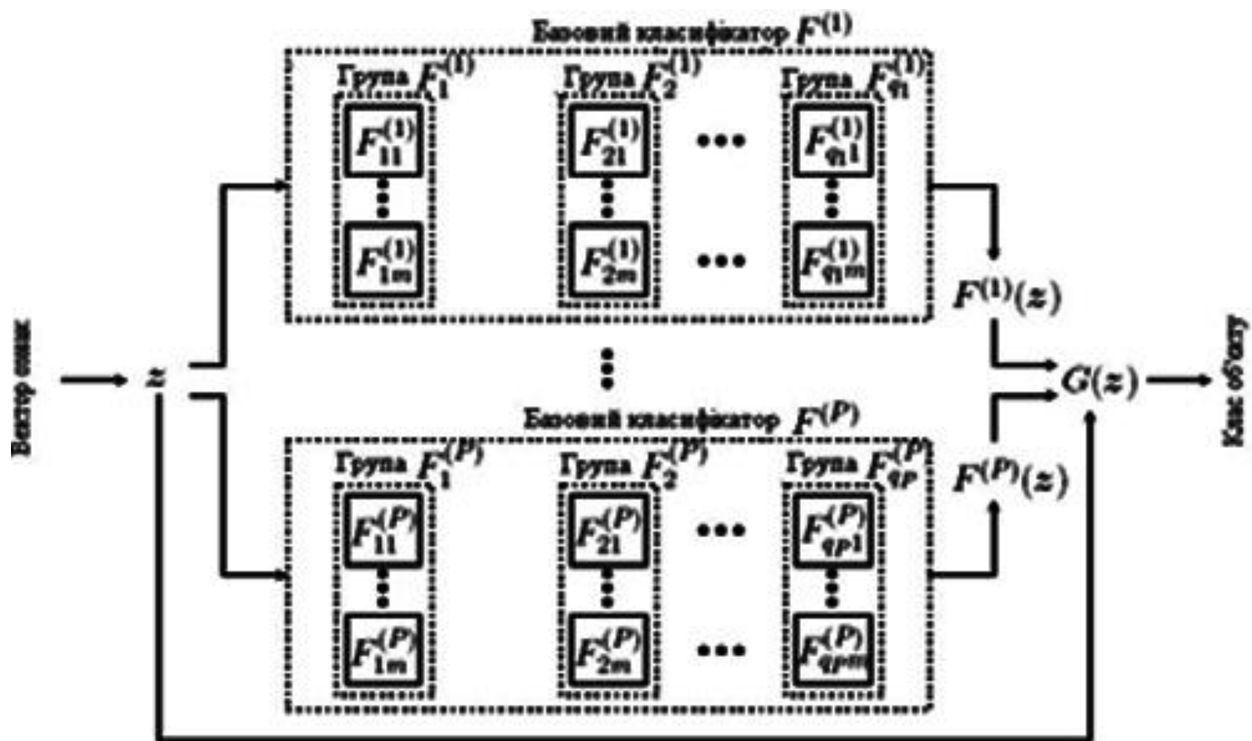



Рис. 4. Схема об'єднання детекторів у багатокласову модель

Об'єднання груп у класифікатор здійснюється на основі гібридного правила, яке представляє собою суміш голосування більшістю й голосування max-wins:

$$F^{(i)}(z) = \left\{ \bar{c} \mid \sum_{j=1}^{q_i} [\bar{c} \in F_j^{(i)}(z)] > \frac{1}{2} \cdot q_i \wedge \Xi_i(\bar{c}) = \max_{\bar{c} \in \{0,\dots,m\}} \Xi_i(\bar{c}`) \right\}_{\bar{c}=0}^{m} \quad (6)$$

Для усунення даного недоліку запропоновано механізм вирішення таких конфліктних випадків, що входять до складу груп класифікатора.

Для побудови колективного правила (агрегація композиції), що об'єднує вихідні результати класифікаторів, були реалізовані такі підходи:

– метод мажоритарного голосування (ММГ), або метод голосування більшістю;

– метод зваженого голосування (МЗГ), що приписує класифікаторам вагові коефіцієнти;
– метод багатоярусного укладання (М<У), доповнений уведенням номера кластера;
– метод Фікса-Ходжеса (МФХ), що представляє собою об'єднання класифікаторів з використанням арбітра на основі динамічних областей компетентності й методу найближчих сусідів.

Для виконання завдання синтаксичного аналізу був реалізований інтерпретатор, що підтримує операції умовного розгалуження, конкатенації класифікаторів, векторного підсумовування, покомпонентного множення й ділення.

У процесі роботи інтерпретатора перевіряється коректність оброблюваного конфігураційного файлу й незапочатковано поля об'єктів всередині споруджуваного дерева класифікаторів. За рахунок використання такої структури в рамках запропонованої методики, стає можливим будувати багаторівневі схеми, у яких блоки будуть містити схожий фрагмент з рис. 4.

Дана методика передбачає розподілену архітектуру, реалізуючи її системи, у яких збір даних здійснюється вторинними вузлами-сенсорами, а вся обробка агрегованих потоків даних виконується на централізованому сервері-колекторі. Другий етап методики, що виконується на стороні сенсорів, полягає в застосуванні розробленого алгоритму збірки сирих пакетів у мережеві з'єднання, виділення їх параметрів і виконанні сигнатурного аналізу з використанням декількох розроблених паралельно модифікацій алгоритмів шаблонного пошуку підрядка. З цією метою було досліджено швидкодію алгоритмів Ахо-Корасік і Бойера-Мура на обраних сигнатурних записах Snort та реалізовано їх поліпшені аналоги за допомогою технологій OpenMP і CUDA. Було реалізовано подієвоорієнтований аналізатор мережевого трафіку, за допомогою якого було вилучено 106 мережевих параметрів, серед яких можна назвати тривалість з'єднання, використовувану мережеву службу, інтенсивність відправки хостом спеціальних пакетів, число активних сполук між конкретною парою IP-адрес (один із критеріїв DoS-атак), ознака зміни масштабування TCP-вікна після фактичного встановлення сесії, поточний стан TCP-з'єднання, різні ознаки наявності скануючих пакетів на рівнях TCP, UDP, ICMP і IP (15 різних кодів сканування). Класифікація цих параметрів показана на рис. 5.




Рис. 5. Класифікація мережевих параметрів

Для вимірювання величини інтенсивності відправки/прийому пакетів використовувався адаптований метод ковзаної середньої. Суть методу полягає у розбитті заданого тимчасового інтервалу, протягом якого проводиться безперервне спостереження за рядом параметрів, на кілька дрібніших інтервалів однакової довжини, початок кожного з яких має зсув 0 <6 відносно початку попереднього інтервалу (рис. 6). $\bigcup_{i=0}^{k-1}\Delta_{\delta\cdot i}^{L`} \subseteq \Delta_0^{(L)}$ і $\bigcup_{i=0}^{k-1}\Delta_{\delta\cdot i}^{L`} \supseteq \Delta_0^{(L)}$, тому $k = 1 + \left\lfloor \frac{L - L`}{\delta} \right\rfloor$. Протягом проміжків часу робляться зліпки значень 0,...,1 параметрів, а їх середня величина (інтенсивність) у рамках тимчасового вікна довжини рядка розраховується за формулою $\overline{\omega} = \frac{1}{k}\cdot\sum_{i=0}^{k-1}\omega_i)$. Тут використовувався інтервал із значенням параметра, рівного п'яти секундам. Довжина згладжуючого інтервалу була обрана рівною одній секунді. Зсув було встановлено у півсекунди. Передбачається, що подібний підхід дозволяє усунути рідкісні по частоті й випадкові мережеві сплески і, тим самим, знизити число помилкових спрацьовувань.

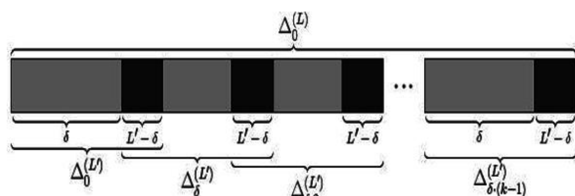



Рис. 6. Ілюстрація методу ковзної середньої

Третій етап методики починається з перевірки сенсорних сигналів, переданих пакетами по протоколу RPC/SSL і містять обчислені параметри з'єднань. Для забезпечення взаємодії колектора й сенсорів вибір упав у бік саме такої зв'язки протоколів, оскільки вони гарантують швидку й безпечну відправку даних. RPC добре зарекомендувала себе технологією, що успішно пройшла випробування часом, і яка дозволяє без праці організувати компактну передачу бінарних потоків даних. SSL, у свою чергу, широко використовується для створення шифрованого каналу між передавачами даних.

Перед безпосереднім навчанням детекторів виконується попередня обробка даних параметрів для зменшення ефекту їх сильної мінливості. Багато методів, включаючи нейронні мережі й метод головних компонентів, чутливі до такого роду флуктуацій і вимагають, щоб усі ознаки оброблюваних векторів мали однаковий масштаб.

Четвертий етап методики, з точки зору обчислювальних ресурсів, є найбільш трудомістким і складається з наступних рекурсивно повторюваних послідовностей дій: обчислення залежностей поточного класифікатора, формування вхідних сигналів для поточного класифікатора, навчання поточного класифікатора. Була розроблена спеціальна деревоподібна структура для зберігання класифікаторів, яка дозволяє здійснювати ефективний спадний спуск по всіх ланцюжках залежностей, починаючи з верхньорівневого класифікатора до термінальних вузлів, представлених детекторами. Навчання кожного класифікатора породжує запит на навчання класифікаторів, що лежать нижче, зазначених у списку його залежностей і генерацію їх вихідних даних для формування вхідних даних вищеописаного класифікатора. Наслідком використовуваного таким чином каскадного навчання є можливість «ледачого» завантаження класифікаторів: у навчанні і розпізнаванні беруть участь тільки ті класифікатори, які безпосередньо чи опосередковано зустрічаються в списку залежностей класифікатора, відповідального за формування спільного рішення у колективі класифікаційних правил. Ця властивість є особливо вигідною при розборі динамічних правил навчання класифікаторів, правил, від успішного або неуспішного спрацьовування яких залежить виклик іншого правила. Зокрема, це характерно для класифікаційного дерева, коли правила є вкладеними одне в одне. Тим самим, за рахунок застосування прийому «ледачого» завантаження вдається уникнути випадків безкорисного виклику того детектора, чиє вихідне значення, як уже відомо, не вплине на результат загального колективу класифікаційних правил.

П’ятий етап методики включає в себе два режими: режим оцінки ефективності й режим функціонування. У першому режимі здійснюється обчислення показників оцінки якості класифікаційних моделей, у другому – виконується діагностика системи без апріорного знання про фактичні класи ідентифікованого з’єднання з мережею.

Розроблена система генерації мережевих атак складається з двох незалежних компонентів:

– асинхронного прозорого проксі-сервера TCP-сесій;
– генератора мережевих атак і frontend-інтерфейсу до нього.

Кожен із названих компонентів націлений на генерацію різноманітних мережевих атак, які можуть включати можливі способи виконання ухилень від сигнатурних мережевих СВА на різних рівнях моделі OSI й являти собою аномальну мережеву активність, яка характеризується скануванням хостів або виходить з ладу в обслуговуванні хоста. Перший компонент призначений для тестування сигнатурних мережевих СВА з метою перевірки їх спроможності до виявлення атак з приховуванням і зі вставкою.

Генератор мережевих атак дозволяє користувачу виконувати певні команди, які спрямовані на формування аномальних мережевих пакетів і їх потоків з метою подальшої їх відправки на IP-адресу жертви й збереження в файл для подальшого детального вивчення (рис. 7).

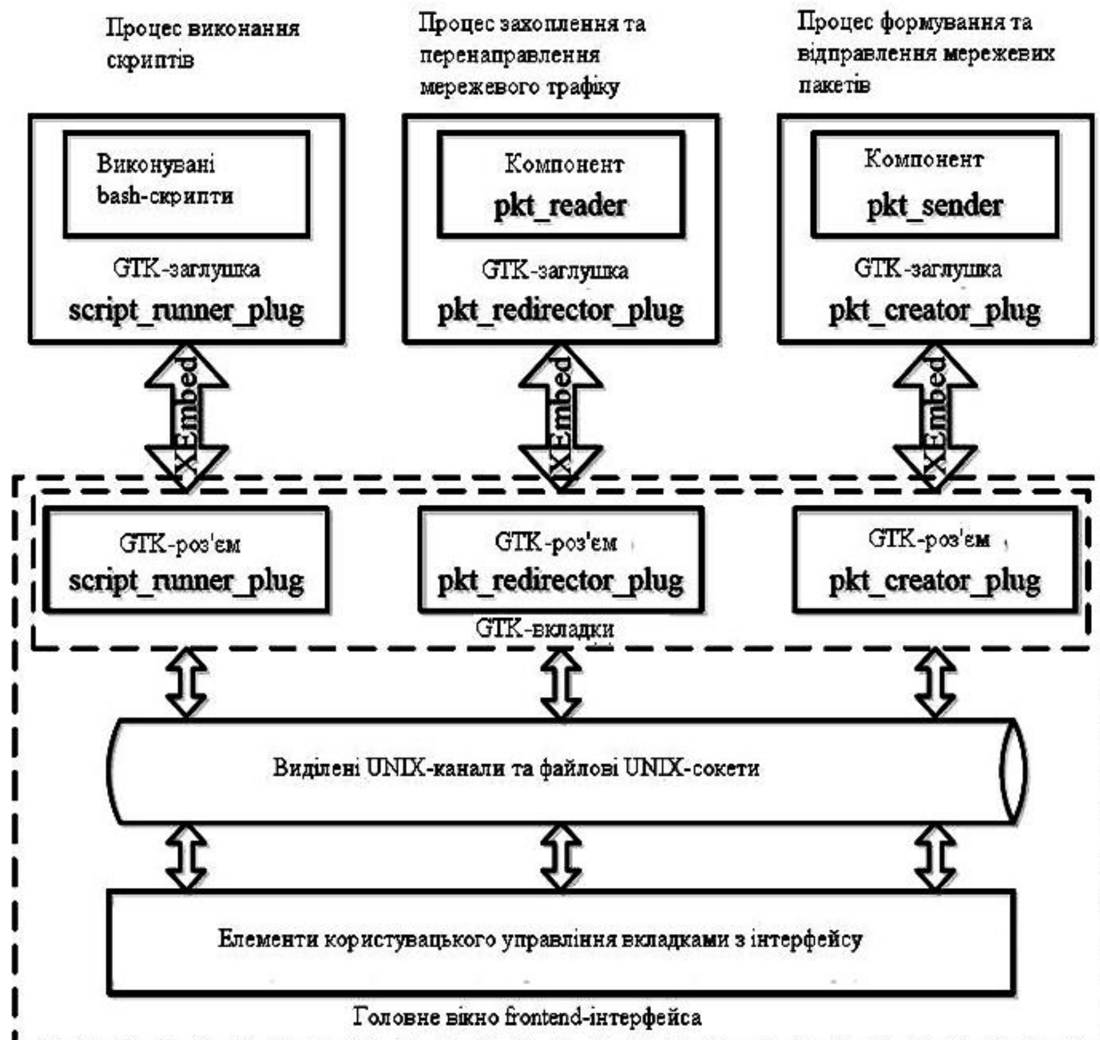



Рис. 7. Архітектура генератора мережевих атак

Для забезпечення відмови стійкості даного генератора атак була розроблена спеціальна архітектура, яка дозволяє виконувати незалежно одразу кілька процесів, які здійснюють ті чи інші операції з мережевими пакетами.

Генератор мережевих атак складається з трьох компонентів: набору виконуваних скриптів, компоненту pkt_reader, компоненту pkt_sender. Представлені компоненти – це бінарні й скриптові виконувані файли, які мають множину вхідних параметрів і запускаються з Linux-консолі. Для надання більш зручного доступу до основних функцій цих компонентів був розроблений frontend-інтерфейс, усередині якого кожен компонент може бути запущений як самостійний процес в окремій Gtk-заглушці, причому в деяких копіях. Перевага такого підходу полягає в тому, що навіть після аварійного завершення відповідного процесу через наявність у його коді критичної помилки, наприклад, помилки сегментації (segmentation fault), основне вікно програми й інші, раніше запущені в рамках нього процеси, залишаться працездатними, і, як наслідок, генерація мережевого трафіку мережевих атак не перерветься в несподіваному місці. У той же час, породження нових процесів призводить до збільшення обсягу споживаної пам'яті, проте, для сучасної обчислювальної техніки, яка має великі апаратні ресурси, створення декількох додаткових процесів не позначається дуже негативно на продуктивності системи. Вбудовування обгорнутих у Gtk-заглушки процесів здійснюється за допомогою створених у головному вікні Gtk-гнізд, взаємодія яких з Gtk-заглушками на системному рівні здійснюється за допомогою протоколу XEmbed. Самі Gtk-гнізда розміщуються в основному вікні програми як вкладки віджета Gtk :: Notebook, який дозволяє додавати нові створені таким чином процеси або перемикатися між уже запущеними процесами. Оскільки кожна вкладка і основне вікно представляють собою різні процеси, то необхідно передбачити механізм їх взаємодії між процесами. У даному програмному засобі в ролі такої зв'язки задіяні так звані Unix-канали та файлові Unix-сокети, ініціалізовані через системні виклики mkfifo і socket (конкретний тип та ім'я файлу як вхідні опції frontend-інтерфейсу). Для асинхронного відстеження атак відкритих дескрипторів каналів і гнізд використовується послідовність Glib-бібліотечних викликів g_io_channel_unix_new і g_io_add_watch, яка дозволяє додати callback-функцію й зчитувати в ній дані з дескрипторів в основному циклі обробки сигналів, що є аналогом виклику слотів. Ця функція викликається тільки при появі в каналі або сокеті нових ще не прочитаних даних. Конкретний склад вкладок формується динамічно з меню й вказується як вхідний аргумент frontend-інтерфейсу. За рахунок цього досягається важлива особливість програмного засобу: додавання, редагування або знищення будь-якої Gtk-заглушки, що не впливає на вихідний код frontend-інтерфейсу, і тим самим ці компоненти можуть розроблятися як самостійні модулі. На даний момент підтримуються три типи вкладок:

ScriptRunnerPlug (рис. 8). Дана вкладка призначена для запуску бінарних і скриптових виконуваних файлів, що генерують аномальний або нормальний трафік. Основні функції включають IP-адресу й порт відправника, IP-адресу й порт одержувача. Вони відображають результат перехоплення потоків кожної із запущених команд sd stdout і stderr шляхом «сліпого» копіювання через текстовий буфер у нижню частину інтерфейсу.

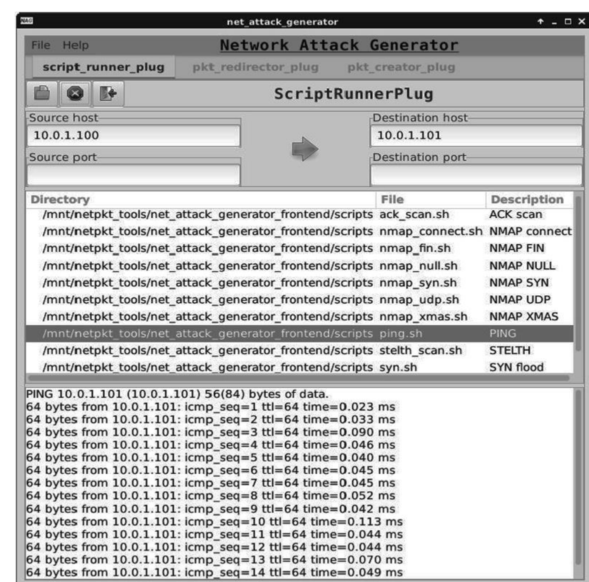



Рис. 8. Вкладка запуску скриптів генератора мережевих атак

PktRedirectorPlug (рис. 9). Дана вкладка призначена для відтворення мережевого трафіку з pcap-файлу або перенаправлення трафіку з одного мережевого інтерфейсу в інший. Серед основних опцій є можливість збереження мережевих дампів, BPF-фільтр і зміни IP-адрес у захоплених пакетах з автоматичним перерахунком контрольних сум на IP-, TCP-, UDP- і ICMP-рівнях.

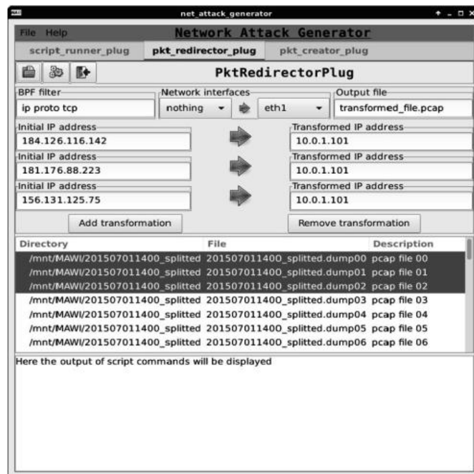



Рис. 9. Вкладка читання мережевих дампів генератора мережевих атак

PktCreatorPlug (рис. 10). Дана вкладка призначена для створення IP-пакетів версії 4. Можлива установка будь-яких полів пакету, починаючи з канального рівня й закінчуючи рівнем призначених для користувача даних. Серед додаткових опцій можна назвати можливість відправки пакета в заданий мережевий інтерфейс, вказівки числа дублів, автоматичного обчислення контрольних сум у пакеті.

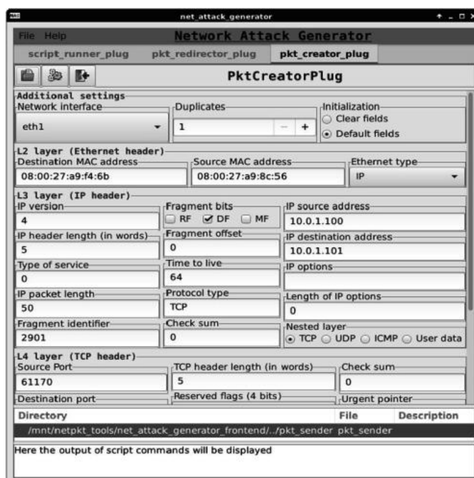



Рис. 10. Вкладка створення мережевих пакетів генератора мережевих атак

Кожна з представлених заглушок – це бінарний виконуваний ELF-файл, який може бути запущений з панелі меню, розташованої у верхній частині frontend-інтерфейсу. Заглушка вбудовується в основне вікно програми за допомогою виклику add_id з аргументом ідентифікатора заглушки над гніздом елементу. При розробці frontend-Інтерфейсу використовувалися мова C++, бібліотеки boost (libboost_program_options.so, libboost_regex.so, libboost_filesystem.so, libboost_system.so, libboost_thread.so) і графічна бібліотека Gtk-3. Обсяг вихідного коду генератора мережевих атак – 2770 непустих рядків без урахування коментарів. За рахунок наявності великої кількості системно залежних викликів у нижчих компонентах і бібліотеках розроблено програмний засіб, призначений тільки для запуску в ОС Linux. Вихідний код представлений у наступних файлах:

– net_attack_gen.h і net_attack_gen.cpp. Файли з описом і реалізацією класу NetAttackGenWindow (основного вікна програми) є похідними від Gtk :: Window;
– base_plug.h і base_plug.cpp. Файли з описом і реалізацією абстрактного базового класу BasePlug (контейнера Gtk-заглушок) є похідними від Gtk :: Plug;
– script_runner_plug.h і script_runner_plug.cpp. Файли з описом і реалізацією класу ScriptRunnerPlug, похідного від BasePlug;
– pkt_redirector_plug.h і pkt_redirector_plug.cpp. Файли з описом і реалізацією класу PktRedirectorPlug, похідного від BasePlug;
– pkt_creator_plug.h і pkt_creator_plug.cpp. Файли з описом і реалізацією класу PktCreatorPlug, похідного від BasePlug;
– interprocess_communicator.h і interprocess_communicator.cpp. Файли з описом і реалізацією класу InterProcessCommunicator підтримуються іменованими каналами, файловими сокетами і виділеною пам'яттю;
– log.h. Файл з директивами для виведення налагоджувальної інформації;
– Makefile. Файл з правилами для збірки додатку.

Кожен з розглянутих алгоритмів реалізовано з використанням мови програмування C++ та бібліотеки libc.so. Для реалізації паралельних модифікацій цих алгоритмів були використані компілятор gcc 4.9.2-10, deb-пакети libgomp1 4.9.2-10 (OpenMP), nvidia-cuda-dev 6.0.37-5, nvidia-cuda-toolkit 6.0.37-5 (CUDA).

**Висновки**

Представлено моделі бінарних класифікаторів, описано алгоритм класифікації мережевих з'єднань за допомогою них.

Розроблено систему генерації мережевих атак. Дана система складається з двох компонентів: асинхронного прозорого проксі-сервера TCP-сесій і frontend-інтерфейсу генератора мережевих атак.

Виконано експерименти з оцінки запропонованих моделей, алгоритмів і методики, проведено експериментальне порівняння розробленого мережевого сенсора з характеристиками відкритих СВА. Результати експериментів підтвердили те, що функціональні і нефункціональні вимоги, а також вимоги, що пред'являються до обчислювальних інтелектуальних систем, виконуються для розробленої СВА.

## Література


1. Refaeilzadeh P., Tang L., Liu H. (2009). *Cross-validation.* Encyclopedia of database systems. Springer, 532–538. DOI: 10.1007/978-0-387-39940-9_565.
2. Riedmiller M., Braun H. (1993). *Direct adaptive method for faster backpropagation learning: The RPROP algorithm.* In Proceedings of IEEE International Conference on Neural Networks. IEEE. DOI: 10.1109/ICNN.1993.298623.
3. Rødfoss J. T. (2011). *Comparison of open source network intrusion detection systems*. The University of Oslo, Department of Informatics, 85.
4. Saied A., Overill R. E., Radzik T. (2016). *Detection of known and unknown DDoS attacks using Artificial Neural Networks.* Neurocomputing. Vol. 172. 385–393. DOI: 10.1016/j.neucom.2015.04.101.
5. Sanders C., Smith J. (2013). *Applied network security monitoring: collection, detection, and analysis*. Elsevier, 496.
6. Scholkopf B., Smola A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press, 626.
7. Thirumalai C., Kar H. (2017). *Memory efficient multi-key (MEMK) generation scheme for secure transportation of sensitive data over cloud and IoT devices.* Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, 1-6. DOI: 10.1109/IPACT.2017.8244948.
8. Al-Ayyoub M., Jararweh Y., Daraghmeh M. (2015). *Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure.* Cluster Comput, Vol. 18, 919–932. DOI: 10.1007/s10586-015-0449-5.
9. Venticinque S., Tasquier L., Di Martino B. (2012). *Agents based cloud computing interface for resource provisioning and management*. Sixth International Conference of the Complex, Intelligent and Software Intensive Systems (CISIS), 249–256. DOI: 10.1109/CISIS.2012.139.
10. Belej O., Halkiv L. (2020). *Development of a network attack detection system based on hybrid neuro-fuzzy algorithms*. CEUR Workshop Proceedings, Vol. 2608, 926-938.


## References


1. Refaeilzadeh P., Tang L., Liu H. (2009). *Cross-validation.* Encyclopedia of database systems. Springer, 532–538. DOI: 10.1007/978-0-387-39940-9_565.
2. Riedmiller M., Braun H. (1993). *Direct adaptive method for faster backpropagation learning: The RPROP algorithm.* In Proceedings of IEEE International Conference on Neural Networks. IEEE. DOI: 10.1109/ICNN.1993.298623.
3. Rødfoss J. T. (2011). *Comparison of open source network intrusion detection systems*. The University of Oslo, Department of Informatics, 85.
4. Saied A., Overill R. E., Radzik T. (2016). *Detection of known and unknown DDoS attacks using Artificial Neural Networks.* Neurocomputing. Vol. 172. 385–393. DOI: 10.1016/j.neucom.2015.04.101.
5. Sanders C., Smith J. (2013). *Applied network security monitoring: collection, detection, and analysis*. Elsevier, 496.
6. Scholkopf B., Smola A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press, 626.
7. Thirumalai C., Kar H. (2017). *Memory efficient multi-key (MEMK) generation scheme for secure transportation of sensitive data over cloud and IoT devices.* Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, 1-6. DOI: 10.1109/IPACT.2017.8244948.
8. Al-Ayyoub M., Jararweh Y., Daraghmeh M. (2015). *Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure.* Cluster Comput, Vol. 18, 919–932. DOI: 10.1007/s10586-015-0449-5.
9. Venticinque S., Tasquier L., Di Martino B. (2012). *Agents based cloud computing interface for resource provisioning and management*. Sixth International Conference of the Complex, Intelligent and Software Intensive Systems (CISIS), 249–256. DOI: 10.1109/CISIS.2012.139.
10. Belej O., Halkiv L. (2020). *Development of a network attack detection system based on hybrid neuro-fuzzy algorithms*. CEUR Workshop Proceedings, Vol. 2608, 926-938.


*Стаття надійшла до редакції 25.06.2020*
*Після доробки 30.07.2020*